Computer-Aided Design 127 (2020) 102857

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

A feature-preserving framework for point cloud denoising

Zheng Liu [a], Xiaowen Xiao [b], Saishang Zhong [c,*], Weina Wang [d], Yanlei Li [b], Ling Zhang [e], Zhong Xie [a]

[a] School of Geography and Information Engineering, National Engineering Research Center of Geographic Information System, China University of Geosciences, Wuhan 430074, China
[b] School of Geography and Information Engineering, China University of Geosciences, Wuhan 430074, China
[c] School of Earth Resources, State Key Laboratory of Geological Processes and Mineral Resources, China University of Geosciences, Wuhan 430074, China
[d] Department of Mathematics, Hangzhou Dianzi University, Hangzhou 310018, China
[e] School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430081, China

ARTICLE INFO

Article history:
Received 26 March 2020
Accepted 16 April 2020

Keywords:
Point cloud denoising
Feature preserving
Second order regularization
Tensor voting
RANSAC

ABSTRACT

Point cloud denoising has been an attractive problem in geometry processing. The main challenge is to eliminate noise while preserving different levels of features and preventing unnatural effects (such as over-sharpened artifacts on smoothly curved faces and cross artifacts on sharp edges). In this paper, we propose a novel feature-preserving framework to achieve these goals. Firstly, we newly define some discrete operators on point clouds, which can be used to construct a second order regularization for restoring a point normal field. Then, based on the filtered normals, we perform a feature detection step by a bi-tensor voting scheme. As will be seen, it is robust against noise and can locate underlying geometric features accurately. Finally, we reposition points with a multi-normal strategy by using a simple yet effective RANSAC-based algorithm. Intensive experimental results show that the proposed method performs favorably compared to other state-of-the-art approaches.

1. Introduction

Point cloud data has attracted considerable attention due to the rapid development of scanning devices (e.g., Microsoft Kinect, Xtion Pro, Google Project Tango, and Intel RealSense). However, even with high-fidelity devices, the corruption of scanned data is usually inevitable from the degradation during its acquisition. Thus, recovering high quality point clouds from the noisy inputs is a typical inverse problem in geometry processing. The main challenge is to distinguish sharp features and noise as they are of high frequency information, and at the same time prevent unnatural effects during the denoising process.

Over the years, state-of-the-art methods have been proposed for recovering noise-free point clouds. In [1], Öztireli et al. proposed a method combining moving least square and local kernel regression to preserve geometric features. However, when the noise level increases, it is unlikely to perform well with satisfactory results. Especially, this limitation is more severe for point clouds containing sharp features. By using robust principal component analysis, [2] exploits sparse characteristics of subspaces to preserve sharp features, which is also robust against outliers. Unfortunately, [2] usually over-sharpens curved features because it depends on sparse and low-rank modeling. In [3], Huang et al. developed an edge-aware upsampling technique to recover the point cloud with well-preserved sharp features. Nevertheless, it sometimes flattens fine details, since it chooses large neighborhood size to pull points away from sharp features for upsampling. Moreover, this method may be sensitive to high density noise, because of its unfaithful upsampling around sharp features without explicitly estimating the point normal field. Sun et al. [4] extended ℓ_0 minimization to point clouds for preserving sharp features. Although their method achieves impressive results for data with piecewise constant priors, it tends to flatten smoothly curved regions and produce pseudo edges in these regions due to its high requirement of sparsity.

As we have seen, the aforementioned state-of-the-art methods are either less able to preserve sharp features well, or may produce unnatural artifacts in denoised results. To overcome these limitations, we propose a novel feature-preserving framework to recover point normals as well as positions. It consists of three cascaded stages: normal filtering; feature detection; and multi-normal point updating, as illustrated in Fig. 1. In the first stage, we present an anisotropic second order variational method to restore the normal field from the noisy input; see Figs. 1(a) and 1(b). Based on the filtered normals, we define normal and point voting tensors, and then introduce a bi-tensor voting scheme to

* Corresponding author.
E-mail address: cugsaishang@foxmail.com (S. Zhong).

https://doi.org/10.1016/j.cad.2020.102857
0010-4485/© 2020 Elsevier Ltd. All rights reserved.

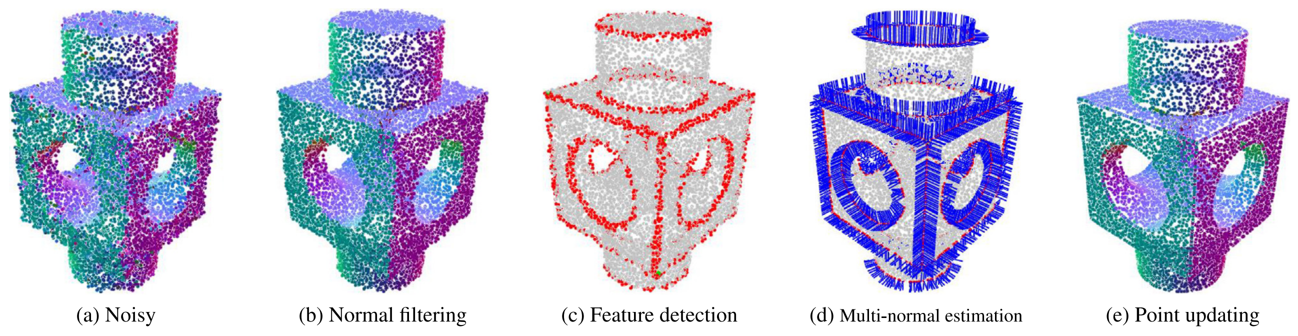

Fig. 1. An overview of the proposed point cloud denoising method. From left to right: noisy input, normal filtering result produced by second order regularization, detected feature points produced by bi-tensor voting scheme, estimated multiple normals at feature points, the final reconstruction result produced by multi-normal point updating.

detect features; see Fig. 1(c). The scheme utilizes the best properties of the normal and point tensor voting and overcomes the weakness of both. In the following multi-normal point updating stage, we estimate the multiple normals per feature point using a RANSAC-based algorithm; see Fig. 1(d). Then, we update point positions according to the restored normals (the normals at non-feature points are produced by our second order normal filtering, and those at feature points are estimated by the RANSAC-based algorithm); see Fig. 1(e). The main contributions of the paper are summarized as follows:

- An anisotropic second order regularization method is presented to restore the point normal field. It is able to preserve sharp features well and simultaneously prevent unnatural effects in denoised results.
- A bi-tensor voting scheme, which combines the normal and point tensor voting, is proposed to detect features on the noisy input. The combined technique is not only robust against noise but also can accurately locate features.
- A simple yet effective RANSAC-based algorithm is introduced to estimate the multiple normals at each feature point. It can significantly reduce cross artifacts during the point reconstruction process.

2. Related work

Point cloud denoising is a fundamental problem in digital geometry processing, which has been studied extensively. It is beyond our scope to review numerous existing methods, and we only review noticeable ones closely related to this work. Generally, we classify point cloud denoising methods into four main categories as following.

MLS-based methods. Moving least squares (MLS) has been originally designed for surface reconstruction by Alexa et al. [5]. Later, many extensions and modifications [6–9] were proposed. As we know, the MLS-based methods iteratively project the input points onto the approximated underlying surface. However, these methods assume the underlying surface with piecewise smooth priors, which lead to blur geometric features inevitably. To overcome this limitation, Öztireli et al. [1] proposed a method combining moving least squares and local kernel regression to project points in a feature-preserving manner. However, their method is less able to preserve sharp features well.

LOP-based methods. Locally optimal projection (LOP) aims at producing a set of points to describe the underlying surface while enforcing a uniform distribution over the input point set. The LOP operator consisting of a data and repulsion term was first presented by Lipman et al. [10]. Huang et al. [11] proposed a weighted LOP (WLOP), which can produce a more uniformly distributed sampling by modifying the repulsion term with local density. Later, Huang et al. [3] reformulated WLOP with the anisotropic weight to preserve sharp features in an edge-aware upsampling methodology. Soon after, Wu et al. [12] applied a deep point representation to consolidate the point cloud containing large holes and missing regions. Prenier et al. [13] reformulated the data term to be a continuous representation of the point set to speed up the LOP process. Lu et al. [14] proposed a projection method based on Gaussian Mixture Model, called GPF, to automatically preserve features of the underlying surface. Although the LOP-based methods yield uniformly distributed sampling results, they usually cannot produce satisfactory denoised results when the noise increases. Especially, this drawback is more severe for surfaces containing sharp features.

Sparse and low-rank methods. Recently, variational methods have received widespread attention. These methods formulate the denoising process as an optimization problem and seek for a desired solution satisfying the optimization goal. As points belonging to the same region will have similar normals, a pioneering work proposed by Avron et al. [15] employed ℓ_1 regularization to restore the point normal field. Sun et al. [4] extended the ℓ_0 minimization to deal with the underlying surface with the piecewise constant prior. The ℓ_1 [15] and ℓ_0 [4] based regularization methods use the sparsity of first order information to remove noise. Both of them preserve sharp features well, but suffer from the undesired staircase effects in smoothly curved regions, especially for ℓ_0 minimization because of its high sparsity requirement. Mattei and Castrodad [2] used low-rank characteristics of subspaces to preserve sharp features and deal with outliers. Nevertheless, their method, which depends on sparse and low-rank modeling, tends to over-sharpen smoothly curved features. Digne et al. [16] proposed a pliable framework to analyze shapes, by consolidating Local Probing Fields (LPFs) defined in the ambient space around the whole shape. In [17], Chen et al. devised a low-rank matrix recovery model with a graph constraint to preserve various geometric features. However, due to the multi-patch collaborative mechanism, their method seems to be computationally intensive.

Data-driven methods. More recently, several data-driven works [18,19] based on deep learning have attracted our attention. In [18], Boulch and Marlet used Hough transform and voting to create the image structure for deep learning without changing in the CNN framework. Roveri et al. [19] designed CNNs, called PointProNets, with a fully differentiable architecture. After converting unordered points to regularly sampled height maps, their method uses PointProNets for point set consolidation to preserve various geometric features and details. These learning-based methods can effectively remove noise and preserve geometric features. Yet, they are highly dependent on the completeness of the training data set.

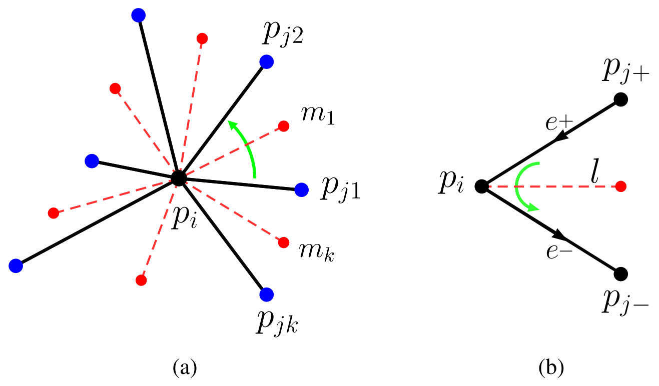

Fig. 2. (a) The illustration of constructing auxiliary geometric elements for the ith point p_i. The blue points are K-nearest neighborhoods (KNNs) of p_i, the red points are midpoints between two consecutive KNNs, the auxiliary edges connecting p_i and its KNNs are plotted in black, and the auxiliary lines connecting p_i and the midpoints are plotted in red. Assume counterclockwise winding order for all geometric elements. (b) the illustration of second order variation over the line l plotted in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3. Normal filtering via second order regularization

The noisy point cloud $\mathcal{P}$ can be considered as a set of unorganized points $\{p_i\}_{i=1}^M$ sampled from a 2-dimensional manifold in $\mathbb{R}^3$, where M is the number of sampled points. For the ith point of $\mathcal{P}$, its index set of K-nearest neighborhood (KNN) is denoted as $\mathcal{N}(i)$, which consists of K elements. We further arrange KNNs of the point in a counterclockwise order using local PCA.

3.1. Definitions of discrete operators on point cloud

To define discrete operators on $\mathcal{P}$, we introduce auxiliary geometric elements to construct local connectivity information for each point. Assume $\mathcal{E} = \{\mathcal{E}(i)\}_{i=1}^M = \{e = (p_i, p_j) : j \in \mathcal{N}(i)\}$ is the set of edges connecting points and their KNNs. For the ith point, the edges of $\mathcal{E}(i)$ are indicated as black lines in Fig. 2(a). Let $\mathcal{L} = \{\mathcal{L}(i)\}_{i=1}^M = \{l = (p_i, m_k)\}$ be the set of lines connecting points and midpoints of their two consistent KNNs. For the ith point, the lines of $\{\mathcal{L}(i)\}$ are indicated as red dash lines in Fig. 2(a).

By constructing the auxiliary edges and lines point by point, we can define discrete operators on $\mathcal{P}$. We denote the vector space $V = \mathbb{R}^M$, which is isomorphic to the discrete function space over $\mathcal{P}$. For any scalar function $u \in V$, the first order difference operator is defined as

$$\mathcal{D}^1 : V \to \underbrace{V \times \cdots \times V}_{K}, \quad u \mapsto \mathcal{D}^1 u,$$

with

$$(\mathcal{D}^1 u)|_e = u_i - u_j, \quad \forall e,$$

where $e = (p_i, p_j)$ and $j \in \mathcal{N}(i)$.

Then, we introduce the isotropic second order operator based on the above first order operator. Given $u \in V$, we define

$$\mathcal{D}^2 : V \to \underbrace{V \times \cdots \times V}_{K}, \quad u \mapsto \mathcal{D}^2 u,$$

with

$$\begin{aligned}(\mathcal{D}^2 u)|_l &= (\mathcal{D}^1 u)|_{e+} - (\mathcal{D}^1 u)|_{e-} \\ &= (u_i - u_{j+}) - (u_{j-} - u_i) \\ &= 2u_i - u_{j+} - u_{j-} \quad \forall l.\end{aligned}$$

For the ith point, we denote the two edges sharing the common point of l as $e- = (p_i, p_{j-})$ and $e+ = (p_{j+}, p_i)$, where p_{j-} and p_{j+} are two consecutive neighborhoods of p_i in counterclockwise order. The aforementioned descriptions are shown in Fig. 2(b).

As we know, compared to the isotropic second order operator, the anisotropic second one should have better feature-preserving property [20,21]. Then, we define the second order operator in an anisotropic manner as

$$\mathrm{D}^2 : V \to \underbrace{V \times \cdots \times V}_{K}, \quad u \mapsto \mathrm{D}^2 u, \tag{1}$$

with

$$\begin{aligned}(\mathrm{D}^2 u)|_l &= w_{e+}(\mathcal{D}^1 u)|_{e+} - w_{e-}(\mathcal{D}^1 u)|_{e-} \\ &= w_{e+}(u_i - u_{j+}) - w_{e-}(u_{j-} - u_i) \\ &= (w_{e+} + w_{e-})u_i - w_{e+}u_{j+} - w_{e-}u_{j-} \quad \forall l,\end{aligned}$$

where w_e is a nonnegative weight function defined on edges.

Furthermore, we extend the above concepts to handle vectorial data. In particular, for a N-channel vectorial field

$$\mathbf{u} = \underbrace{u \times \cdots \times u}_{N} \in \mathbb{R}^{M \times N},$$

the proposed operators can be calculated channel by channel.

3.2. Second order normal filtering

Given a noisy point cloud $\mathcal{P}$, we use the local PCA normal estimation to acquire the initial normals $\bar{\mathbf{n}}$. To remove noise in $\bar{\mathbf{n}}$, we propose the following anisotropic second order normal filtering model

$$\min_{\mathbf{n} \in \mathcal{C}} \left\{ \sum_{l \in \mathcal{L}} \mathrm{len}(l) \|(\mathrm{D}^2 \mathbf{n})|_l\| + \frac{\alpha}{2} \sum_i \mathrm{disk}(p_i) \|\mathbf{n}_i - \bar{\mathbf{n}}_i\|^2 \right\}, \tag{2}$$

where $\mathcal{C} = \{\mathbf{n}_i : \|\mathbf{n}_i\| = 1\}$ and α is a tuning parameter which is empirically set in the range of $[1000, 6000]$. $\mathrm{len}(l)$ is the length of line l, and $\mathrm{disk}(p_i)$ is the area of the circle whose center is p_i and radius is the average length of edges contained in $\mathcal{E}(i)$. Here, the edge weight w_e used in computing the anisotropic second order operator is defined as an exponential function $w_e = \exp(-(\frac{1 - \mathbf{n}_i \cdot \mathbf{n}_j}{1 - \cos(\theta)})^2)$, where $\mathbf{n}_i$ and $\mathbf{n}_j$ are normals of two adjacent points of edge e, and θ is the angle parameter in the range of $[30°, 60°]$.

Because of the nondifferentiability and nonlinear constraints of the problem (2), it is challenging to directly solve it. The studies in [22–24] show that variable-splitting and augmented Lagrangian method (ALM) have achieved great success in solving ℓ_1 related problems. Here, we first introduce an auxiliary variable X and reformulate the problem (2) as

$$\begin{aligned}&\min_{\mathbf{n}, X} \left\{ \sum_l \mathrm{len}(l) \|X_l\| + \frac{\alpha}{2} \sum_i \mathrm{disk}(p_i) \|\mathbf{n}_i - \bar{\mathbf{n}}_i\|^2 + \psi(\mathbf{n}) \right\} \\ &\text{s.t.} \quad X = \mathrm{D}^2 \mathbf{n},\end{aligned} \tag{3}$$

where

$$\psi(\mathbf{n}) = \begin{cases} 0, & \mathbf{n} \in \mathcal{C}, \\ +\infty, & \mathbf{n} \notin \mathcal{C}. \end{cases}$$

The augmented Lagrangian of (3) reads

$$\begin{aligned}L(\mathbf{n}, X; \lambda) =& \sum_l \mathrm{len}(l) \|X_l\| + \frac{\alpha}{2} \sum_i \mathrm{disk}(p_i) \|\mathbf{n}_i - \bar{\mathbf{n}}_i\|^2 + \psi(\mathbf{n}) \\ &+ \sum_l \mathrm{len}(l) \big(\lambda_l \cdot (X_l - (\mathrm{D}^2 \mathbf{n})|_l)\big) + \frac{r}{2} \sum_l \mathrm{len}(l) \|X_l - (\mathrm{D}^2 \mathbf{n})|_l\|^2,\end{aligned} \tag{4}$$

where $\lambda = \{\lambda_l\}$ is the Lagrange multiplier and r is a positive penalty coefficient. The primal variables update procedure can be separated into two subproblems.

(1) $\mathbf{n}$-subproblem: the sub-minimization problem of $\mathbf{n}$ can be written as

$$\min_{\mathbf{n}} \frac{\alpha}{2}\sum_i \text{disk}(p_i)\|\mathbf{n}_i - \bar{\mathbf{n}}_i\|^2 + \frac{r}{2}\sum_l \text{len}(l)\|(\mathrm{D}^2\mathbf{n})|_l - (X_l + \frac{\lambda_l}{r})\|^2 + \psi(\mathbf{n}), \tag{5}$$

which is a quadratic optimization with the unit normal constraints. Here we adopt an approximate strategy to solve this problem. Specifically, we first ignore $\psi(\mathbf{n})$ and solve a quadratic programming and then project the minimizer onto a unit sphere. Here, we use fixed-point iteration to approximately solve the quadratic optimization problem.

(2) X-subproblem: the sub-minimization problem of X is given as

$$\min_X \sum_l \text{len}(l)\|X_l\| + \frac{r}{2}\sum_l \text{len}(l)\|X_l - ((\mathrm{D}^2\mathbf{n})|_l - \frac{\lambda_l}{r})\|^2 \tag{6}$$

This problem is easy to solve because the energy function (6) can be spatially decomposed, where the minimization problem with respect to each l is performed individually. Thus, for each X_l, we only need to solve the following problem

$$\min_{X_l} \|X_l\| + \frac{r}{2}\|X_l - ((\mathrm{D}^2\mathbf{n})|_l - \frac{\lambda_l}{r})\|^2,$$

which has a closed form solution as

$$X_l = \text{shrink}(r,\ (\mathrm{D}^2\mathbf{n})|_l - \frac{\lambda_l}{r}), \tag{7}$$

where $\text{shrink}(v, w) = \max(0, 1 - \frac{1}{v\|w\|})w$.

The entire procedure for solving the variational model (2) is outlined in Algorithm 1. Based on variable-splitting and ALM, this algorithm iteratively solves the above two subproblems and updates the Lagrange multiplier.

Algorithm 1: ALM for solving second order normal filtering model (2)

```
Initialization: $\mathbf{n}^{-1} = 0, X^{-1} = 0, \lambda^0 = 0, k = 0, \varepsilon = 1e-6$;
repeat
    Solve $\mathbf{n}$-subproblem
        For fixed $(\lambda^k, X^{k-1})$, compute $\mathbf{n}^k$ from (5) ;
        Normalize $\mathbf{n}^k$;
    Solve $X$-subproblem
        For fixed $(\lambda^k, \mathbf{n}^k)$, compute $X^k$ from (7) ;
    Update Lagrange multiplier
        $\lambda^{k+1} = \lambda^k + r(X^k - (\mathrm{D}^2\mathbf{n})^k)$ ;
until $\sum_i \text{disk}(p_i)\left\|\mathbf{n}_i^k - \mathbf{n}_i^{k-1}\right\|^2 < \varepsilon$ or $k \geq 150$;
return $\mathbf{n}^k$.
```

4. Feature detection by bi-tensor voting

The tensor voting is a fundamental tool in geometry processing for accurately detecting features on high-quality meshes [25,26]. Recently, it was extended to point clouds by analyzing the normal voting tensor [27,28] or the point voting tensor [29,30]. However, performing direct the normal or point tensor voting on noisy point clouds usually produces spurious effects. Specifically, the point tensor voting is sensitive to the noise, which tends to produce pseudo features in smooth transition regions; see Fig. 3(a). Although the normal tensor voting is less sensitive to the noise, it is prone to detect redundant points around underlying geometric features; see Fig. 3(b). In this paper, we propose a bi-tensor voting scheme combining the normal and point tensor voting to detect features on point clouds. The proposed scheme is robust against noise, while has advantages in locating underlying geometric features accurately; see Fig. 3(c).

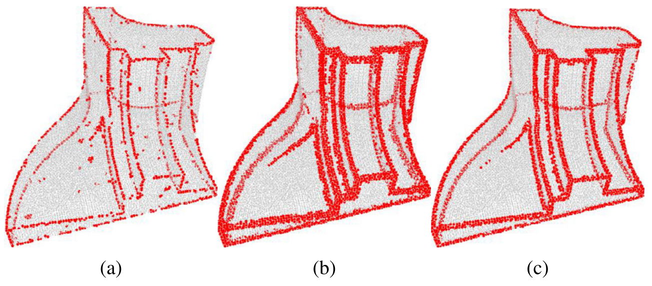

Fig. 3. Feature detection results produced by the (a) point tensor voting, (b) normal tensor voting, and (c) our bi-tensor voting scheme.

4.1. Construction of normal and point voting tensors

With the filtered normals from the previous filtering stage, we construct the normal and point voting tensors simultaneously for the following voting analysis. The normal voting tensor for the ith point is calculated as the sum of weighted covariance matrices from normals of its KNNs

$$\mathrm{T}_i^{\mathbf{n}} = \frac{1}{\sum_{j\in\mathcal{N}(i)} \mathrm{w}^{\mathbf{n}}(p_i, p_j)} \sum_{j\in\mathcal{N}(i)} \mathrm{w}^{\mathbf{n}}(p_i, p_j)(\mathbf{n}_j \otimes \mathbf{n}_j), \tag{8}$$

where $\mathrm{w}^{\mathbf{n}}(p_i, p_j) = \exp(-\|p_i - p_j\|^2/2\sigma_p^2)$ is a Gaussian weight function decreasing in terms of the distance between p_i and p_j. The standard deviation σ_p is empirically fixed as the average distance between points. The symbol $\otimes$ denotes the outer product $\mathbf{n}_j\mathbf{n}_j^T$. Similarly, the point voting tensor is constructed as the sum of weighted covariance matrices from its KNNs

$$\mathrm{T}_i^{p} = \frac{1}{\sum_{j\in\mathcal{N}(i)} \mathrm{w}^{p}(\mathbf{n}_i, \mathbf{n}_j)} \sum_{j\in\mathcal{N}(i)} \mathrm{w}^{p}(\mathbf{n}_i, \mathbf{n}_j)\big((p_j - \bar{p}) \otimes (p_j - \bar{p})\big), \tag{9}$$

where $\mathrm{w}^{p}(\mathbf{n}_i, \mathbf{n}_j) = \exp(-\|\mathbf{n}_i - \mathbf{n}_j\|^2/2\sigma_n^2)$ is a Gaussian weight function decreasing in terms of the variation of two neighboring normals and $\bar{p} = \frac{1}{K}\sum_{j\in\mathcal{N}(i)} p_j$. σ_n is a user specified parameter.

As the above two tensors are symmetric and positive semidefine, they can be diagonalized by eigenvalues and represented in terms of their spectral components. Specifically, the normal voting tensor (8) can be written as

$$\mathrm{T}_i^{\mathbf{n}} = \sum_{m=1}^{3} \lambda_{i,m}^{\mathbf{n}} e_{i,m}^{\mathbf{n}} \otimes e_{i,m}^{\mathbf{n}}, \tag{10}$$

where $\lambda_{i,m}^{\mathbf{n}}$ and $e_{i,m}^{\mathbf{n}}$ are the corresponding eigenvalues and eigenvectors. Assume the eigenvalues are sorted in decreasing order $\lambda_{i,1}^{\mathbf{n}} \geq \lambda_{i,2}^{\mathbf{n}} \geq \lambda_{i,3}^{\mathbf{n}} \geq 0$. Similarly, the point voting tensor (9) can be denoted in terms of its spectral components

$$\mathrm{T}_i^{p} = \sum_{m=1}^{3} \lambda_{i,m}^{p} e_{i,m}^{p} \otimes e_{i,m}^{p}, \tag{11}$$

where $\lambda_{i,m}^{p}$ and $e_{i,m}^{p}$ are the corresponding eigenvalues and eigenvectors with $\lambda_{i,1}^{p} \geq \lambda_{i,2}^{p} \geq \lambda_{i,3}^{p} \geq 0$.

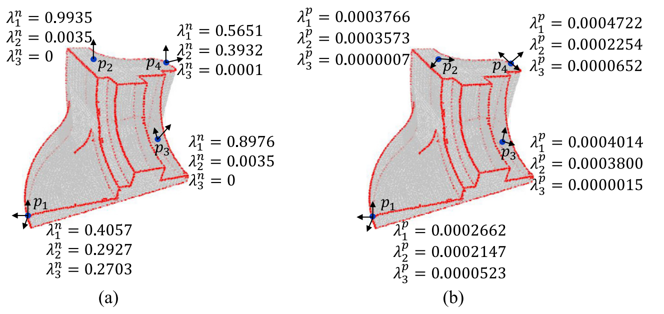
$\lambda_1^n = 0.9935$
$\lambda_2^n = 0.0035$
$\lambda_3^n = 0$
$\lambda_1^n = 0.5651$
$\lambda_2^n = 0.3932$
$\lambda_3^n = 0.0001$
p_2
p_4
p_3
$\lambda_1^n = 0.8976$
$\lambda_2^n = 0.0035$
$\lambda_3^n = 0$
p_1
$\lambda_1^n = 0.4057$
$\lambda_2^n = 0.2927$
$\lambda_3^n = 0.2703$
(a)
$\lambda_1^p = 0.0003766$
$\lambda_2^p = 0.0003573$
$\lambda_3^p = 0.0000007$
$\lambda_1^p = 0.0004722$
$\lambda_2^p = 0.0002254$
$\lambda_3^p = 0.0000652$
$\lambda_1^p = 0.0004014$
$\lambda_2^p = 0.0003800$
$\lambda_3^p = 0.0000015$
$\lambda_1^p = 0.0002662$
$\lambda_2^p = 0.0002147$
$\lambda_3^p = 0.0000523$
(b)

Fig. 4. Eigenvalues of (a) the normal voting tensor and (b) the point voting tensor for different points. Point p_1 lies at the corner, p_2 and p_3 lie on two faces, p_4 lies on the sharp edge.

4.2. *Bi-tensor voting analysis*

In this section, a bi-tensor voting scheme is proposed, which combines the advantages of the normal and point tensor voting. To determine whether a point is a feature or not, we propose a new measure R_f. For the ith point, this measure is defined as

$$\mathrm{R}_{f,i} = \mathrm{R}_{f,i}^{\mathbf{n}} \cdot \mathrm{R}_{f,i}^{p} = \frac{\lambda_{i,2}^{\mathbf{n}}}{\sum_{m=1}^{3} \lambda_{i,m}^{\mathbf{n}}} \cdot \frac{\lambda_{i,3}^{p}}{\sum_{m=1}^{3} \lambda_{i,m}^{p}}. \tag{12}$$

The reasons to define this measure are as follows. $\mathrm{R}_{f,i}^{\mathbf{n}}$ of the point p_i on the face is smaller than that on the edge or at the corner. Similar phenomenon of $\mathrm{R}_{f,i}^{p}$ also can be observed. The multiplication of $\mathrm{R}_{f,i}^{\mathbf{n}}$ and $\mathrm{R}_{f,i}^{p}$ makes $R_{f,i}$ of the feature point (point at the corner or on the edge) is significantly larger than that of the non-feature point (point on the face). The values of R_f in a concrete example are illustrated in Table 1. As can be seen, R_f of p_1, p_2, and p_5 (feature points lie at the corner and on the edges) are obviously larger than a given threshold. Therefore, we can easily classify feature points as

$$\mathcal{P}_f = \{p_i \in \mathcal{P} \mid \mathrm{R}_{f,i} > \epsilon_f\}, \tag{13}$$

where ϵ_f is a feature threshold.

In order to further identify corners, we propose a measure R_c at point p_i as

$$\mathrm{R}_{c,i} = \mathrm{R}_{c,i}^{\mathbf{n}} \cdot \mathrm{R}_{c,i}^{p} = \frac{\lambda_{i,3}^{\mathbf{n}}}{\sum_{m=1}^{3} \lambda_{i,m}^{\mathbf{n}}} \cdot \frac{\lambda_{i,3}^{p}}{\lambda_{i,1}^{p} - \lambda_{i,2}^{p}}. \tag{14}$$

$\mathrm{R}_{c,i}^{\mathbf{n}}$ of p_i on the corner is larger than that on the edge or on the face. Similar phenomenon can be observed as the value of $\mathrm{R}_{c,i}^{p}$. Again, the multiplication of $\mathrm{R}_{c,i}^{\mathbf{n}}$ and $\mathrm{R}_{c,i}^{p}$ makes the change in value of $\mathrm{R}_{c,i}$ be more significant. Actually, if the point p_i lies on the corner, $\mathrm{R}_{c,i}$ should be larger than a given threshold; otherwise, it may lie on the face or edge; see the values of R_c in Table 1 for example. Thus, we can identify the set of features on corners as

$$\mathcal{P}_c = \{p_i \in \mathcal{P} \mid \mathrm{R}_{c,i} > \epsilon_c\}, \tag{15}$$

where ϵ_c is a corner threshold. When there are more than one candidate corners identified in a local neighborhood region, the one with the largest value of R_c is chosen. Consequently, we can use the aforementioned rules (13) and (15) to detect almost all the features and identify the corner points.

To further illustrate the efficiency of the proposed bi-tensor voting scheme, we compare it with the feature detection scheme proposed by Zhang et al. [29]. As we can see in Fig. 5, the method in [29] produces some pseudo features on the smooth regions, and has a few redundant features around the underlying sharp edges. In contrast, the proposed bi-tensor voting scheme gives more satisfying results. It greatly reduces the pseudo features in smooth regions, and at the same time locates the detected features more accurately. The example in Fig. 5 demonstrates that the proposed feature detection scheme is more robust to the perturbation of noise than the method in [29].

Table 1
Values of R_f and R_c for points on Fandisk shown in Fig. 4.

	$\mathrm{R}_f^{\mathbf{n}}$	R_f^{p}	R_f	$\mathrm{R}_c^{\mathbf{n}}$	R_c^{p}	R_c
p_1	0.30216	0.09809	0.02963	0.27903	1.01553	0.2834
p_2	0.00351	0.00095	$3.35 \times e^{-6}$	0	0.03626	0
p_3	0.00388	0.00191	$7.44 \times e^{-6}$	0	0.0701	0
p_4	0.41027	0.08547	0.035	0.0001	0.26418	$2.64 \times e^{-5}$

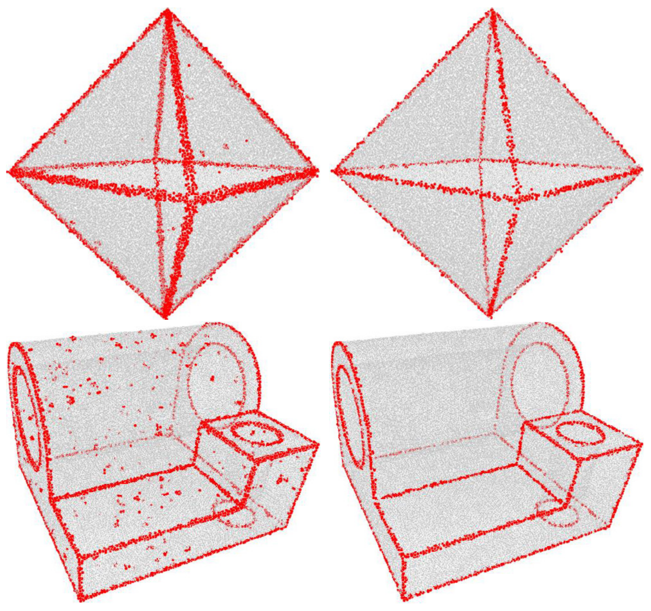

Fig. 5. Feature detection results of two noisy point clouds corrupted by Gaussian noise with standard deviation of 60% and 20% of the average distance between points respectively. From left to right: detected features produced by the method [29] and ours.

Remark. Both thresholds ϵ_f and ϵ_c are sensitive to noise. In our experiments, we empirically set ϵ_f in the range of $[0.01, 0.05]$, and set ϵ_c in the range of $[0.07, 0.2]$.

5. Multi-normal point updating

After obtaining filtered point normals, it is necessary to reposition points to match the filtered normals. Because the single normals at feature points are ambiguous and undefinable, it is necessary to adopt multi-normal point updating strategy as proposed in previous works [30–32]. This technique can overcome cross artifacts at sharp features and preserve sharp features better than the conventional single normal based approaches, especially when the point cloud is corrupted with high level of noise [30]. Hence, we similarly employ the multi-normal technique to design our point updating method consisting of two stages. At the first stage, we propose a RANSAC-based approach to estimate multiple normals at each feature point. Then, we reposition points in a multi-normal manner. Details of both are elaborated as follows.

5.1. *RANSAC-based multi-normal estimation*

With the filtered single point normals, we propose a simple yet effective multi-normal estimation based on RANSAC algorithm. The proposed algorithm repeats a voting process enough times to find a good solution with high probability.

Algorithm 2: RANSAC-based multi-normal estimation for a feature point p_i.

```
Initialization: $P = \hat{\mathcal{N}}(i), j = 0, m = 0, S = 0, S_{max} = 0$;
repeat
    $m = 0$;
    $S_{max} = 0$;
    repeat
        (1) $X \leftarrow$ SelectThreeCoplanarPoints($P$);
        $S = area(X)$;
        if $S \geq S_{max}$ then
            $S_{max} = S$;
            (2) $A \leftarrow$ BuildPlane(X);
        end
    until $m < 20$;
    (3) $Y_{i,j} \leftarrow$ GroupPointsBelongToPlane($P$, $A$);
    (4) $\mathbf{n}_{i,j} \leftarrow$ GenerateNormalVector($Y_{i,j}$);
    (5) $P \leftarrow$ ExcludePointsYFromP($P$, $Y_{i,j}$);
until $size(P) \leq 4$ or $j > 6$;
```

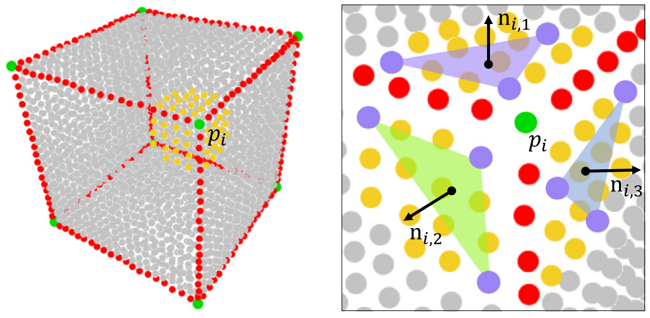

Fig. 6. An example for demonstrating our RANSAC-based multi-normal estimation algorithm.

For the ith point, our algorithm estimates multiple normals of it using non-feature points in its KNNs denoted as $\hat{\mathcal{N}}(i)$. Since $\hat{\mathcal{N}}(i)$ may contain points on several different surface patches, we partition $\hat{\mathcal{N}}(i)$ into d_i groups $\{Y_{i,j}\}_{j=1}^{d_i}$ and estimate the corresponding multiple normals $\{\mathbf{n}_{i,j}\}_{j=1}^{d_i}$, where d_i represents the number of normals estimated for point p_i. By the way, we have $d_i = 1$ for all the non-feature points. We use an example in Fig. 6 to demonstrate our algorithm, where edge points, the corner point, and non-feature points are colored in red, green, and yellow respectively. To estimate multiple normals at corner point p_i, we iteratively perform the following steps

(1) Selecting three coplanar non-feature points as a set X. We randomly select three non-feature points with similar normals (the angles between these normals are smaller than 15°). In practice, we pick one non-feature point at random, and select the other two non-feature points together to form a triangle as large as possible. As shown in Fig. 6, the selected points are colored in purple.

(2) Building the plane A from X (using the average point and normal of X). Approximately, the plane A can be regarded as an underlying surface nearing the feature point p_i.

(3) Grouping points belonging to the plane A. We first project all the neighbors in $\hat{\mathcal{N}}(i)$ to the plane. If the projection distance of a point is less than half of average distance between points, we consider the point belonging to the plane. Here, we use $Y_{i,j}$ to denote the points belonging to the plane A.

(4) Generating normal vector $\mathbf{n}_{i,j}$. We directly generate a normal vector by averaging all the normals at the set $Y_{i,j}$.

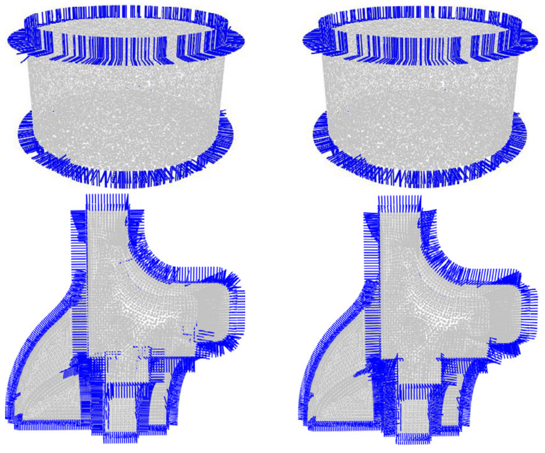

Fig. 7. Multi-normal estimation results at feature points. From left to right: results produced by (a) method in [30] and (b) our RANSAC-based algorithm.

(5) Excluding points $Y_{i,j}$ from non-feature points $\hat{\mathcal{N}}(i)$. If the number of the left points is large than 4, the iteration process goes to the step (1); otherwise, the above process is terminated and returns the estimated multiple normals at p_i.

The overall procedure is outlined in Algorithm 2. By iteratively running the above steps for each feature point, we can estimate multiple normals at each feature point.

In general, the multiple normals at each edge point should include two normal vectors, while those at corner point should have at least three normal vectors. However, when estimating the multiple normals at edge point nearing corner point, the proposed algorithm may produce redundant normal vectors. To address this problem, we further introduce a simple score metric for each normal vector, which is designed as:

$$score(\mathbf{n}_{i,j}) = \min_{p \in Y_{i,j}} \|p_i - p\|_2. \tag{16}$$

This *score* can be used to measure the accuracy of the normal vector. Generally, the smaller the *score* is, the more accurate the corresponding normal vector is. Thus, if the number of normal vectors of an edge point is larger than 3, we only select the normal vectors with two smallest scores as the multiple normals of the edge point.

5.2. Comparisons and examples of multi-normal estimation

To evaluate our multi-normal estimation, we compare it with the latest one proposed by Zhang et al. [30]. We employ the error metric $RMSM_\tau$ (Root Mean Square measure with threshold) designed in [30] to estimate the angular error between the multiple normals of the ground truth and those of the estimated result. The ground-truth data used in the paper are point clouds with multiple normals at each point, which are provided by Zhang et al. [30]. Based on the error metric and ground-truth data, we perform the following comparisons. The visual comparisons are shown in Fig. 7, and the corresponding quantitative comparisons are listed in Table 2. As we can see, $RMSM_\tau$ errors of our method are close to those of the method [30]. However, our method is much faster than the compared one. To further demonstrate the validity of our multi-normal estimation, we perform it on a series of point clouds with different shapes; see Fig. 8. It is clear to see that, our multi-normal estimation scheme can accurately produce multiple normals at almost each feature point for the tested point clouds.

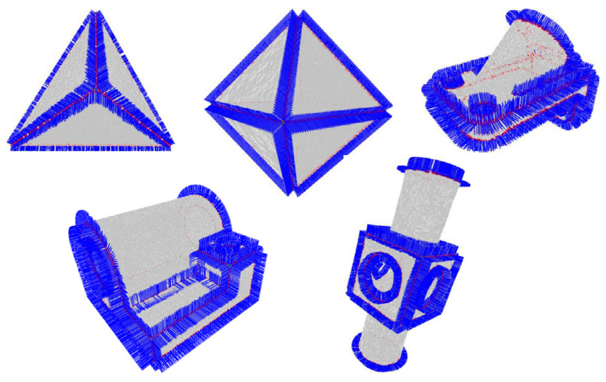

Fig. 8. Multi-normal estimation results of our RANSAC-based algorithm for different point clouds.

Table 2
Quantitative Comparisons of multi-normal estimation results shown in Fig. 7.

	$RMSM_\tau$	Time (in s)		
	Method [30]		Ours	
Cylinder	0.42	347	0.42	82
Fandisk	10.7	265	10.5	39

Table 3
Quantitative comparisons of denoising results produced by methods RIMLS, MRPCA, EAR, ℓ_0, and ours.

	$D_{mean} \times 10^{-3}$, $A_{mean} \times 10^{-2}$				
	RIMLS	MRPCA	EAR	ℓ_0	Ours
Block	5.48, 13.7	5.35, 8.25	8.69, 8.54	5.61, 8.18	**5.06**, **7.21**
Dodecahedron	8.50, 8.83	7.40, 2.12	8.90, 5.19	6.41, **0.73**	**6.37**, 0.78
Bunny-iH	3.47, 5.90	3.65, 6.35	3.36, 5.97	3.98, 6.34	**2.86**, **5.80**
Armadillo	1.45, 9.61	1.33, 10.9	1.40, 11.0	1.35, 9.97	**1.30**, **9.25**
Boy	7.75, 13.5	8.44, 15.8	7.98, 13.2	8.04, 14.4	**7.63**, **13.1**
Nicolos	5.82, 7.05	6.42, 8.20	5.98, 7.21	6.07, 7.51	**5.66**, **6.88**

5.3. Point positions updating

With the filtered normals at non-feature points and the estimated multiple normals at feature points, we need to update point positions to match these normals. According to the multi-normal strategy [30,31], we reposition points by solving the following minimization problem:

$$\min_{\mathcal{P}} \sum_{i=1}^{M} \frac{1}{d_i} \sum_{r=1}^{d_i} \sum_{j \in \mathcal{N}(i)} \sum_{s=1}^{d_k} \frac{\mathbf{w}(i,r,j,s)}{d_k \mathcal{W}_{ij}} [\mathbf{n}_{j,s} \cdot (p_i - p_j)]^2, \tag{17}$$

where the weight $\mathbf{w}(i,r,j,s) = \mathrm{w}^p(n_{i,r}, n_{j,s})\mathrm{w}^{\mathbf{n}}(p_i, p_r)$, and $\mathcal{W}_{ij} = \sum_{j \in \mathcal{N}(i)} \sum_{s=1}^{d_k} \frac{\mathbf{w}(i,r,j,s)}{d_k}$ is a normalization factor. The definitions of $\mathrm{w}^p(n_{i,r}, n_{j,s})$ and $\mathrm{w}^{\mathbf{n}}(p_i, p_r)$ can be found in Eqs. (9) and (8) respectively. By using gradient descent to solve the minimization problem (17), we update the point positions by the following iterative formula

$$p_i^{m+1} = p_i^m + \frac{1}{d_i} \sum_{r=1}^{d_i} \sum_{j \in \mathcal{N}(i)} \sum_{s=1}^{d_k} \frac{\mathbf{w}(i,r,j,s)}{d_k \mathcal{W}_{ij}} [\mathbf{n}_{j,s} \cdot (p_i^m - p_j^m)] \mathbf{n}_{j,s}. \tag{18}$$

More details can refer to the work [31].

6. Experimental results and comparisons

To testify the performance of our point cloud denoising method, we perform it on both synthetic and real data. The tested point clouds are contaminated by either synthetic or raw scanned noise. The synthetic noise is generated by a zero-mean Gaussian function with standard deviation proportional to the diagonal of the axis-aligned bounding box of the ground truth. We also provide visual and quantitative comparisons of our method to the state-of-the-art ones including RIMLS [1], MRPCA [2], EAR [3] and ℓ_0 [4]. For the method EAR, we perform the code provided by its authors; for the other three methods, we have implemented them according to their published articles by using C++. All of the examples are run on a laptop with a Intel i7 core 2.6 GHz processor and 8GB RAM.

6.1. Experimental setup

To objectively evaluate the performances of the tested methods, we utilize the following configuration items. Specifically, we first apply PCA estimation to produce the same initial normals for all the tested methods, and then use the bilateral filter in [3] to restore the same normal field for MRPCA and EAR. Similarly to [14,17], we sometimes upsample the results of the tested methods, and reconstruct the upsampled results via RIMLS (provided by Meshlab for feature-preserving reconstruction), for enhancing visual effects. Besides, back-face culling is also adopted for visual rendering. We carefully tune the parameters of each tested method to yield satisfactory results.

6.2. Qualitative comparisons and examples

Fig. 9 demonstrates comparisons of a point cloud containing both sharp features and smoothly curved regions. As we can see, all of the tested methods are able to remove noise. Furthermore, RIMLS recovers smooth regions well but over-smoothes sharp features seriously. MRPCA and EAR not only blur sharp features to some extent, but also generate staircase effects in smooth transition regions, see Figs. 9(c) and 9(d). Although MRPCA preserves sharp features better than EAR, it may cause shape shrinkage and distortion. Due to the sparse property of ℓ_0 norm and ℓ_1 norm, both ℓ_0 and our method preserve sharp features well, see Figs. 9(e) and 9(f). However, due to its high sparsity requirement, ℓ_0 flattens some smooth regions and produces false edges in these regions. In contrast, our method can preserve sharp features and simultaneously recover smooth transition regions well. As a result, visual comparisons in Fig. 9 show that our method is noticeably better than the other compared methods in terms of recovering sharp features and smooth regions.

Fig. 10 shows results on a point cloud containing only flat regions and sharp features. As we can see, ℓ_0 and our method can effectively remove noise while preserving sharp features well, see Figs. 10(e) and 10(f). In contrast, MRPCA and EAR do a good job on flat regions, but they blur sharp features in varying degrees; see Figs. 10(c) and 10(d). Again, RIMLS smoothes sharp features evidently for removing noise; see Fig. 10(b). This example shows the effectiveness of our method for dealing with the point cloud with piecewise constant priors.

Fig. 11 gives comparisons on a point cloud with rich details. As we can see, ℓ_0 flattens some details and smooth regions. EAR and MPRCA sharpens some smoothly curved features, and MRPCA makes this situation even worse for its sparse property; see Figs. 11(d) and 11(c). Besides, RIMLS and our method both produce visually satisfactory results; see Figs. 11(b) and 11(f). Nevertheless, from numerical errors listed in Table 3, we observe that errors of our method are always lower than those of RIMLS. Thus, for the point cloud with different levels of features, our method also produces the appealing result with more geometric features than the other methods.

Fig. 12 demonstrates comparisons on a point cloud containing multi-scale features. Obviously, our method and all the compared

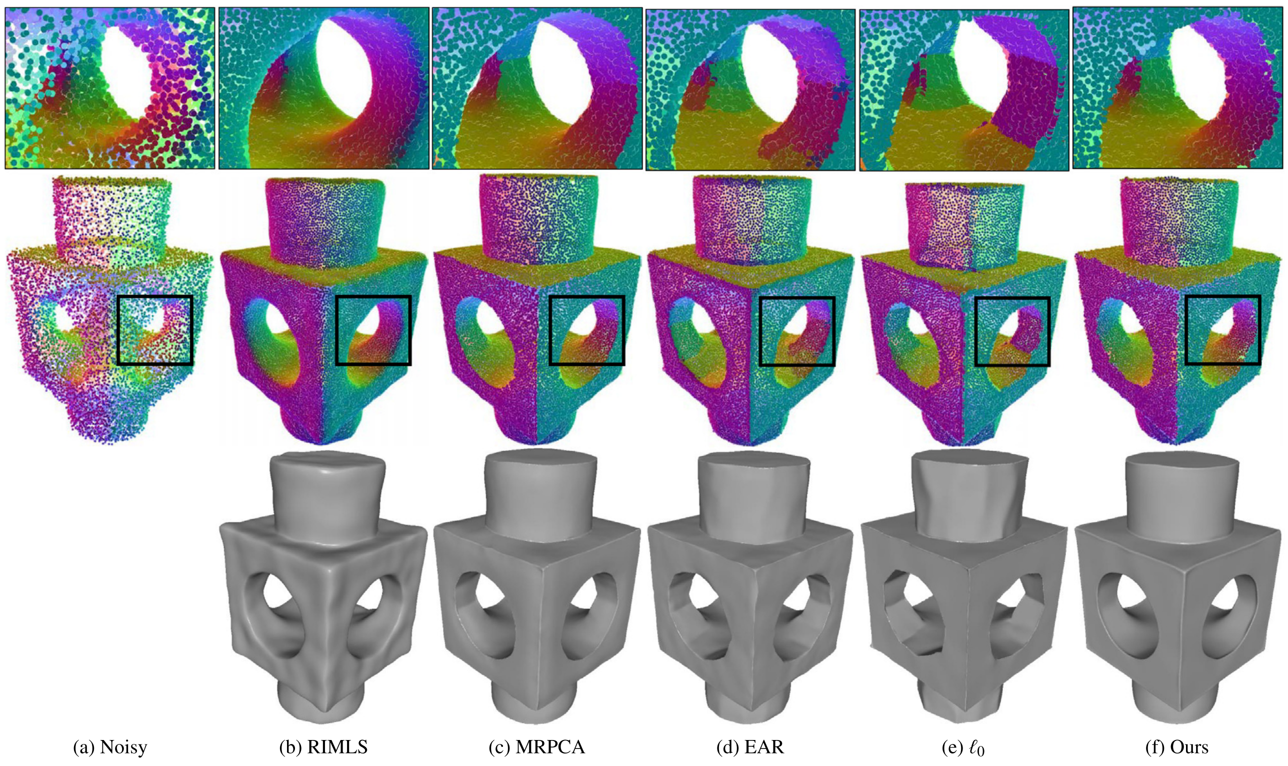

Fig. 9. Denoising results of Block ($M = 20.0$ K, 1.5% noise). From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method, respectively. The third row shows the corresponding surface reconstruction results. The zoomed views, showed in the first row, highlight that our method better preserves sharp features and smoothly curved regions.

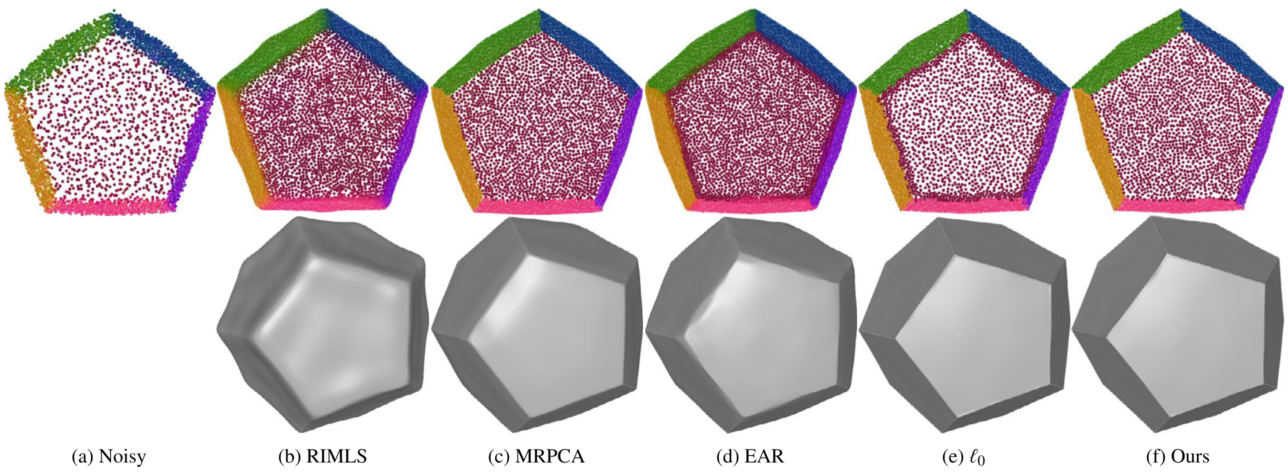

Fig. 10. Denoising results of Dodecahedron ($M = 10.8$ K, 1% noise). From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method, respectively. The second row shows the corresponding surface reconstruction results.

methods can effectively remove noise. However, RIMLS smoothes small-scale features; see the eye regions in Fig. 12(b). EAR preserves large-scale features well, but it sharpens some small-scale features; see Fig. 12(d). Due to the sparsity requirements, MRPCA and ℓ_0 flatten some smooth regions; see the nose regions in Figs. 12(c) and 12(e). In contrast, our method outperforms the other methods in terms of recovering multi-scale features; see Fig. 12(f).

To further testify the validity of our method, we test it on two laser scanned point clouds, shown in Figs. 13 and 14. As we can see in Fig. 13(b), although RIMLS recovers smooth regions well, it over-smoothes some structure features. In contrast, MRPCA, EAR, and ℓ_0 keep the structures, but they inevitably sharpen smoothly curved regions, see Figs. 13(c), 13(d) and 13(e). Due to the good properties of the proposed second order operator, our method can restore the piecewise smooth structures well. Similarly, MRPCA and EAR sharpen some fine details, while ℓ_0 flattens these details; see Figs. 14(c), 14(d), and 14(e). As we can see in Fig. 14(b), RIMLS over-smoothes fine details at small scales. In contrast, our method preserves fine details better than the compared methods; see the zoomed views of Fig. 14.

Recently, more and more point clouds are acquired by Kinect sensors. To evaluate the effectiveness of our method on Kinect scanning data, we show the comparison results on this type of data in Fig. 15. As we can see, RIMLS, MRPCA, and ℓ_0 induces artifacts to some extent. In contrast, EAR and our method both yield visually better denoising results; see Figs. 15(d) and 15(f). However, from the quantitative comparisons in Table 3, we can

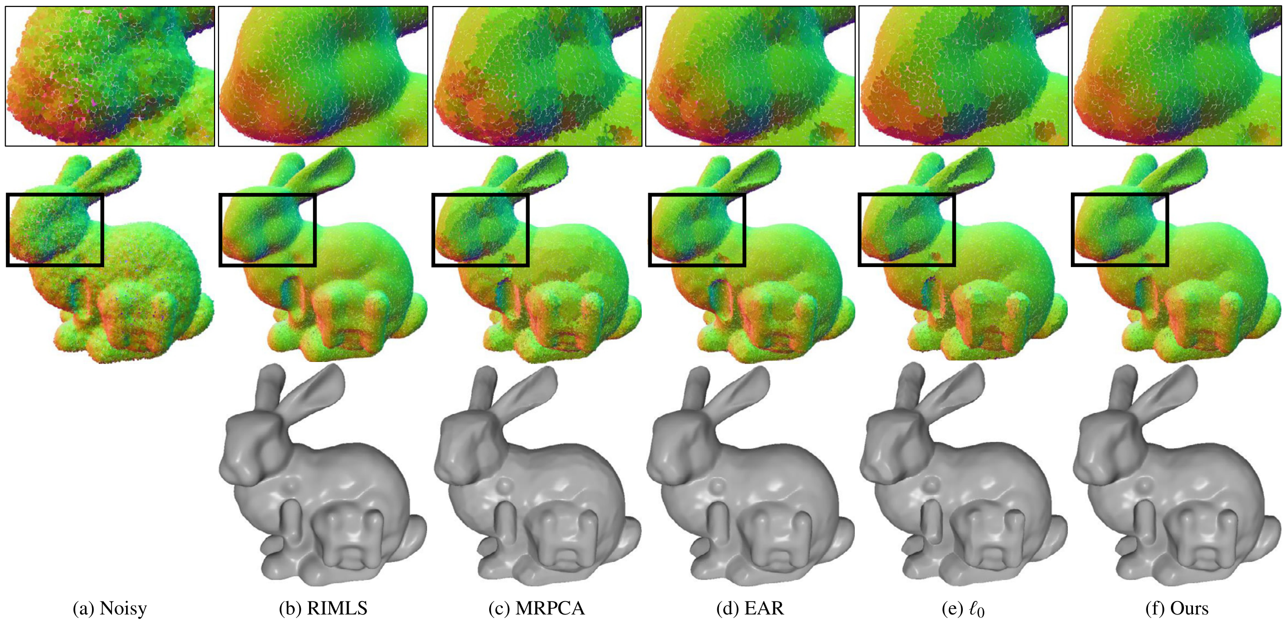

Fig. 11. Denoising results of Bunny-iH (M = 110.5 K, 1.2% noise). From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method, respectively. The third row shows the corresponding surface reconstruction results. The zoomed views, showed in the first row, highlight that our method better preserves different levels of features and prevents unnatural effects.

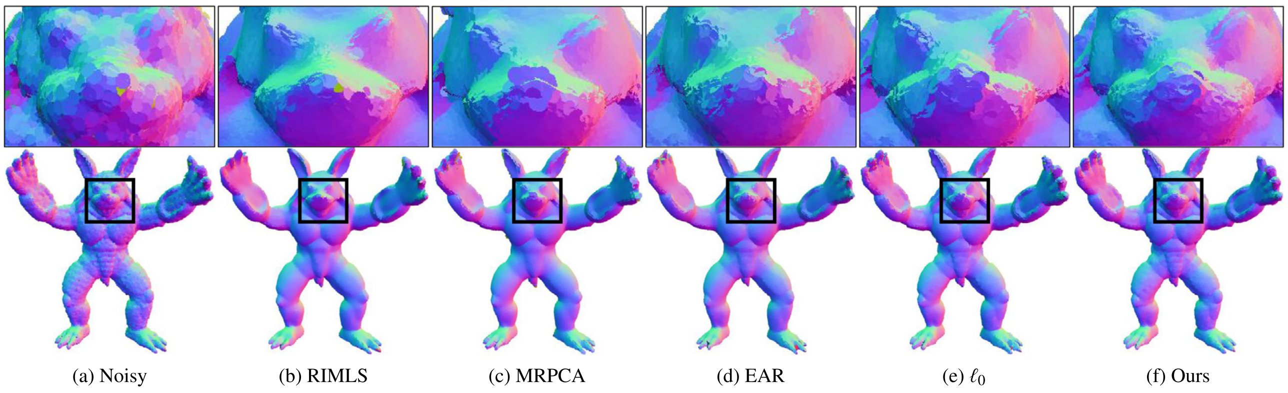

Fig. 12. Denoising results of Armadillo (M = 80 K, 0.5% noise). From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method. The zoomed views, showed in the first row, highlight that our method better preserves multi-scale geometric features.

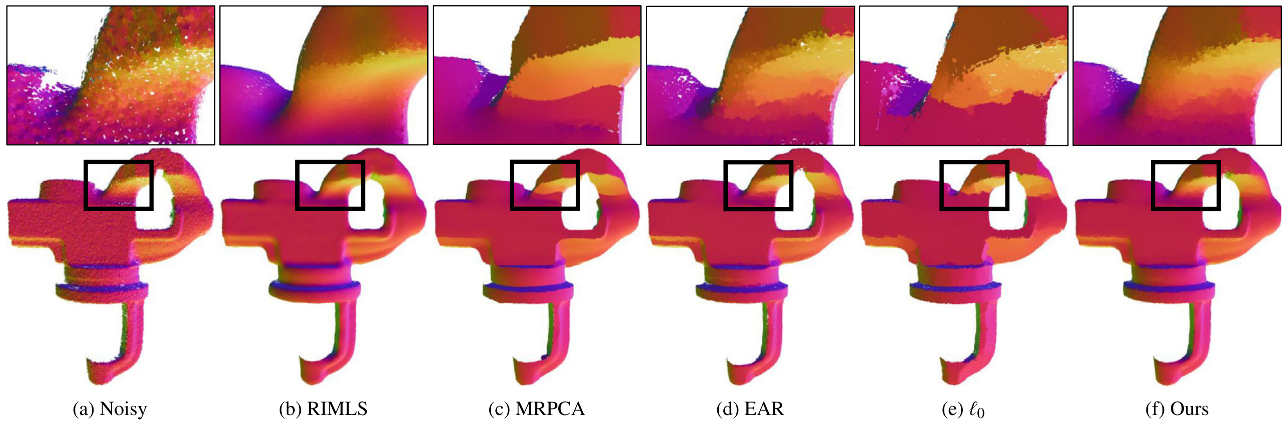

Fig. 13. Denoising results of Iron (M = 161 K). From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method. The zoomed views, showed in the first row, highlight that our method better keeps smoothly curved features.

find that the errors of our method are always lower than those of EAR. Thus, even for the data produced by Kinect, our method still can effectively suppress noise while yield the satisfactory result containing more faithful features.

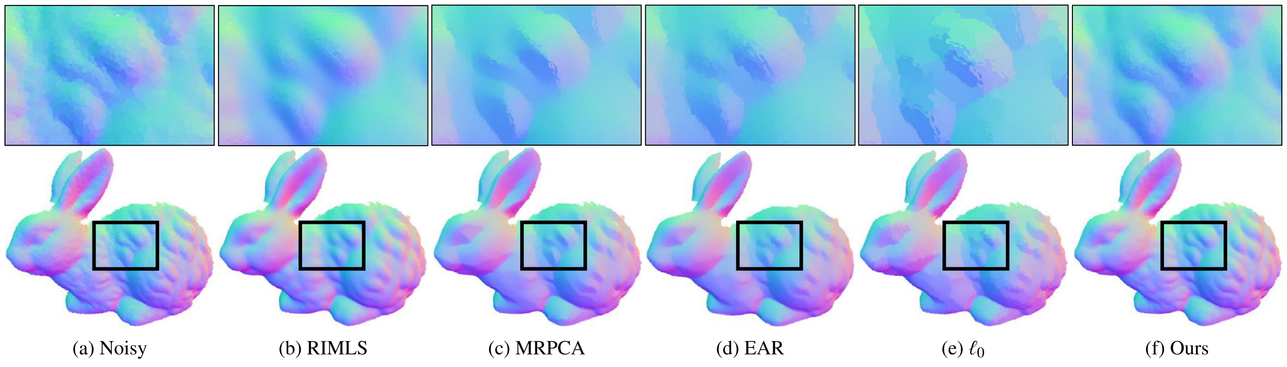

Fig. 14. Denoising results of Rabbit ($M = 37.4$ K). From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method. The zoomed views, showed in the first row, highlight that our method better keeps small-scale features.

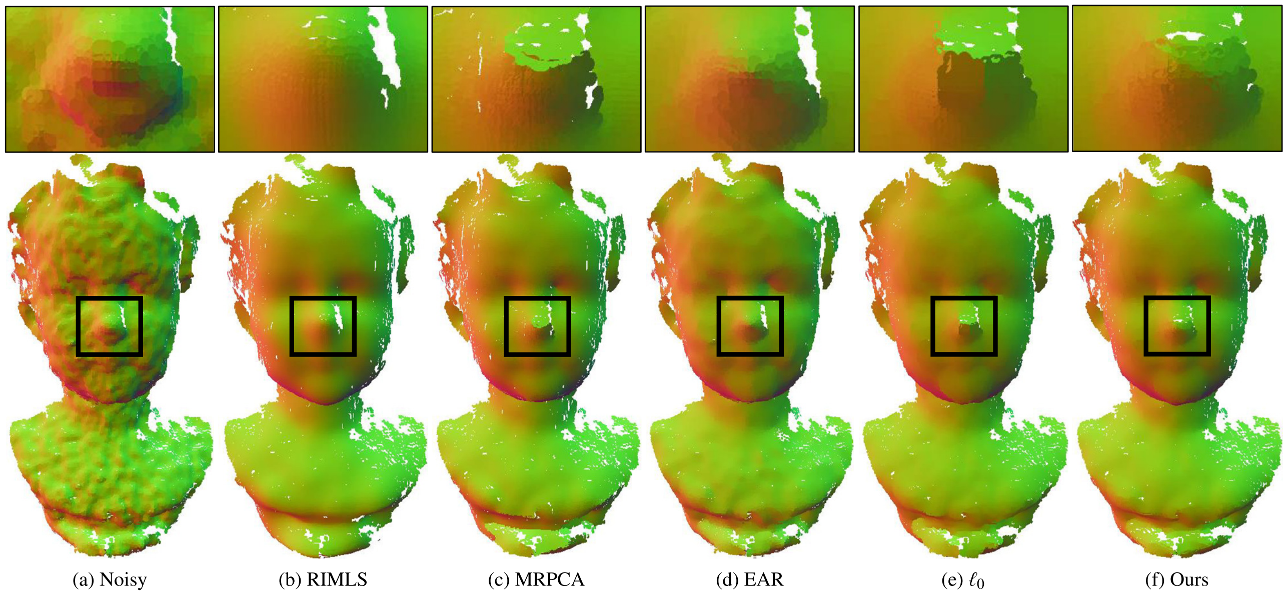

Fig. 15. Denoising results of Boy ($M = 28.2$ K) captured by Kinect. From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method. The zoomed views, showed in the first row, highlight that our method better keeps smoothly curved features.

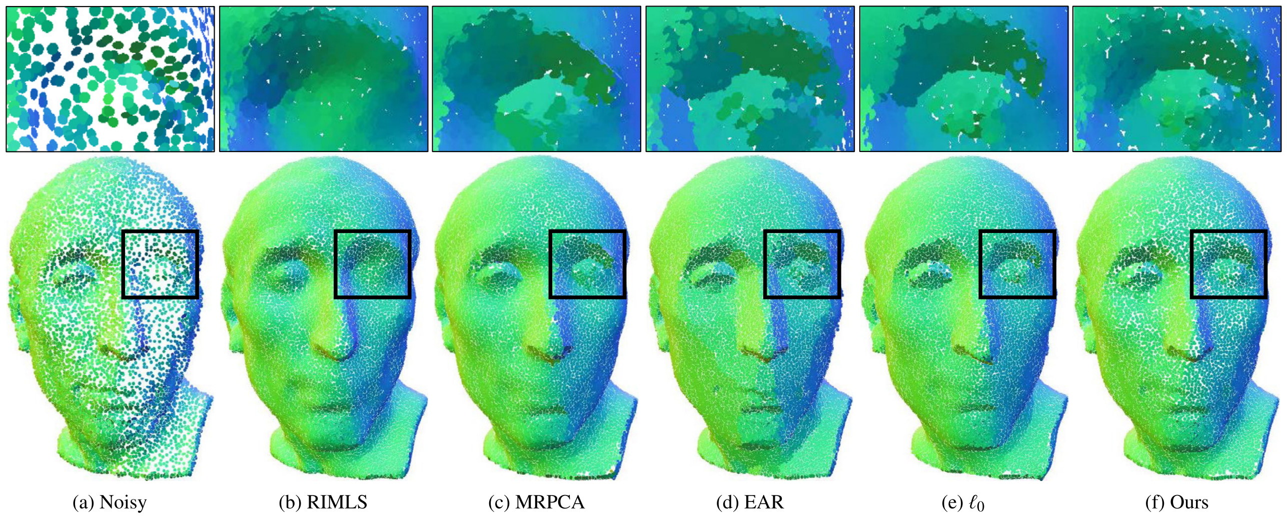

Fig. 16. Denoising results of Nicolos ($M = 14.8$ K, 0.5% noise). From left to right: noisy input, results produced by RIMLS, MRPCA, EAR, ℓ_0, and our method. The first row shows the zoomed views.

We also verify the effectiveness of our method on irregular sampling data. As can be seen in Fig. 16, although the noisy input is of varying density distributions, similar results can be observed as those of the examples in Figs. 11 and 12. There

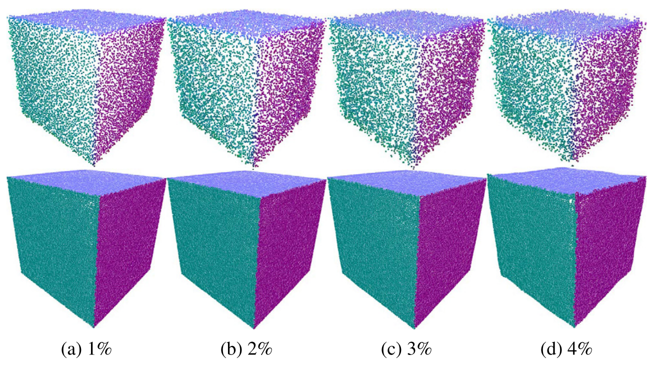

Fig. 17. Denoising results of Cube corrupted by different levels of noise. The first row shows noisy point clouds (1%, 2%, 3%, and 4% noise), while the second row shows the corresponding denoising results produced by our method.

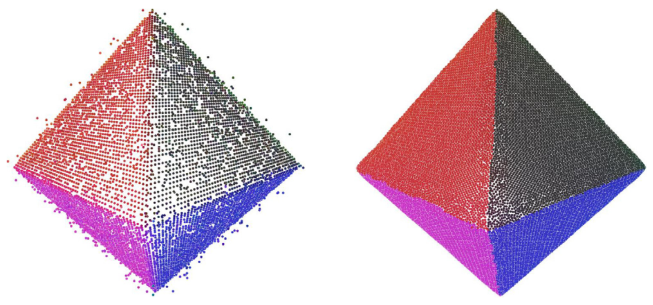

Fig. 18. Denoising result of Octahedron with outliers. Noisy input (left) and the corresponding denoising result (right).

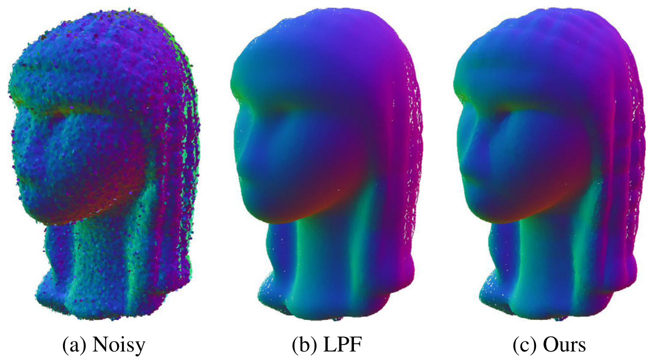

Fig. 19. Denoising results of Brassempouy. From left to right: noisy input, results produced by the method in [16] (LPF) and ours.

are over-smoothed effects existing in the result of RIMLS, and over-sharpen effects existing in those of MRPCA, EAR and ℓ_0. In contrast, our method again produces the more faithful result, which shows it is robust against irregular sampling.

Fig. 17 shows the robustness of our method against different levels of noise. As can be seen in Figs. 17(a), 17(b), and 17(c), our method can effectively remove noise while preserving sharp features when the noise level increases. However, when the noise level is too high, our method will fail to produce the satisfactory result; see Fig. 17(d) for example.

Our method also takes outliers into account when processing data with a poor quality. Fig. 18 demonstrates that our method is capable of dealing with outliers.

It is worth to compare our method with the recent one proposed by Digne et al. [16]. They have introduced a framework for point set denoising by consolidating Local Probing Fields in the ambient space. Their framework allows the shape to reveal its non-local similarities, which is good at recovering smooth features of the shape. In contrast, our method optimizes the point normal field using ℓ_1 norm, which has a better feature-preserving property. Thus, our method is more robust for preserving geometric features; see Fig. 19 for example.

Table 4
Running times (in s).

Model	RIMLS	MRPCA	EAR	ℓ_0	Ours
Block (20.0K)	39.7	48.7	7.07	70.6	34.4
Dodecahedron (10.8K)	17.9	24.8	5.12	26.9	20.8
Bunny-iH (110.5K)	331.0	531.1	305.1	392.3	155.6
Armadillo (80K)	225.7	373.5	206.4	286.7	108.5
Iron (161K)	562.7	2602.9	528.6	657.1	226.9
Rabbit (37.4K)	62.9	124.3	49.6	108.8	47.5
Boy (28.2K)	59.6	75.3	20.9	97.2	41.8
Nicolos (14.8K)	25.8	32.5	6.3	45.8	22.1

6.3. *Quantitative comparisons*

The above visual comparisons demonstrate that our method can produce better denoised results than the compared methods. Here, we further quantitatively compare our method to others on synthetic data. We utilize the metric D_{mean} to evaluate position errors, and the metric A_{mean} to evaluate normal errors. The definitions of these two metrics can be found in [31]. We compute D_{mean} and A_{mean} for the examples shown in Figs. 9, 10, 11, 12, 15, and 16, and record these errors in Table 3. As we can observe, the denoising results produced by our method have the least D_{mean} errors in all the cases, and have the least A_{mean} errors in most cases. These show that the results yielded by our method are more faithful to the ground truth point clouds.

We also record CPU costs for all the tested methods in Table 4. As can be seen, for small-scale point clouds, the CPU costs of our method are reasonable. More importantly, for large-scale point clouds, our method is faster than all the other methods; see the CPU costs of Bunny-iH in Fig. 11 and Iron in Fig. 13 which have more than 100 K points.

7. Conclusion

In this paper, a feature-preserving framework has been proposed to recover a noisy-free point cloud. It first utilizes a second order regularization to restore the normal field. With the filtered normals, a well-designed bi-tensor voting scheme is introduced to detect features, which overcomes the weakness of the normal and point tensor voting. Finally, point positions are reconstructed by the multi-normal strategy to reduce cross artifacts. Experimental results show that our denoising method outperforms the state-of-the-art approaches for preserving different levels of geometric features without introducing unnatural artifacts.

Besides the denoising application, we expect to extend our work to handle the wider class of problems, such as point cloud segmentation, surface reconstruction, and urban architecture modeling.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank anonymous reviewers for their constructive suggestions for improving the manuscript. Zheng Liu's research is supported by National Natural Science Foundation of China (No. 61702467). Weina Wang's research is supported by Zhejiang Provincial Natural Science Foundation of China (No. LQ20A010007). Ling Zhang's research is supported by China Postdoctoral Science Found (No. 2018M642933) and National Natural Science Foundation of China (No. 61902286). Zhong Xie's research is supported by National Natural Science Foundation of China (No. 41671400) and National Key Research and Development Program (No. 2018YFB0505500).

References

[1] Öztireli AC, Guennebaud G, Gross M. Feature preserving point set surfaces based on non-linear kernel regression. Comput Graph Forum 2009;28(2):493–501.
[2] Mattei E, Castrodad A. Point cloud denoising via moving RPCA. Comput Graph Forum 2017;36(8):123–37.
[3] Huang H, Wu S, Gong M, Cohen-Or D, Ascher U, Zhang HR. Edge-aware point set resampling. ACM Trans Graph 2013;32(1):9:1–9:12.
[4] Sun Y, Schaefer S, Wang W. Denoising point sets via ℓ_0 minimization. Comput Aided Geom Design 2015;35:2–15.
[5] Alexa M, Behr J, Cohen-Or D, Fleishman S, Silva CT. Computing and rendering point set surfaces. IEEE Trans Vis Comput Graphics 2003;9(1):3–15.
[6] Fleishman S, Cohen-Or D, Silva CT. Robust moving least-squares fitting with sharp features. ACM Trans Graph 2005;24(3):544–52.
[7] Adamson A, Alexa M. Point-sampled cell complexes. ACM Trans Graph 2006;25(3):671–80.
[8] Guennebaud G, Gross M. Algebraic point set surfaces. ACM Trans Graph 2007;26(3):23.
[9] Guennebaud G, Germann M, Gross M. Dynamic sampling and rendering of algebraic point set surfaces. Comput Graph Forum 2010;27(2):653–62.
[10] Lipman Y, Cohen-Or D, Levin D, Tal-Ezer H. Parameterization-free projection for geometry reconstruction. ACM Trans Graph 2007;26(3):22.
[11] Huang H, Li D, Zhang H, Ascher U, Cohen-Or D. Consolidation of unorganized point clouds for surface reconstruction. ACM Trans Graph 2009;28(5):176:1–7.
[12] Wu S, Huang H, Gong M, Zwicker M, Cohen-Or D. Deep points consolidation. ACM Trans Graph 2015;34(6):176:1–3.
[13] Preiner R, Mattausch O, Arikan M, Pajarola R, Wimmer M. Continuous projection for fast ℓ_1 reconstruction. ACM Trans Graph 2014;33(4):47:1–13.
[14] Lu X, Wu S, Chen H, Yeung S-K, Chen W, Zwicker M. GPF: GMM-inspired feature-preserving point set filtering. IEEE Trans Vis Comput Graphics 2017;24(8):2315–26.
[15] Avron H, Sharf A, Greif C, Cohen-Or D. ℓ_1-Sparse reconstruction of sharp point set surfaces. ACM Trans Graph 2010;29(5):135:1–2.
[16] Digne J, Valette S, Chaine R. Sparse geometric representation through local shape probing. IEEE Trans Vis Comput Graphics 2017;24(7):2238–50.
[17] Chen H, Wei M, Sun Y, Xie X, Wang J. Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint. IEEE Trans Vis Comput Graphics 2019.
[18] Boulch A, Marlet R. Deep learning for robust normal estimation in unstructured point clouds. Comput Graph Forum 2016;35(5):281–90.
[19] Roveri R, Öztireli AC, Pandele I, Gross M. PointProNets: Consolidation of point clouds with convolutional neural networks. Comput Graph Forum 2018;37(2):87–99.
[20] Liu Z, Zhong S, Xie Z, Wang W. A novel anisotropic second order regularization for mesh denoising. Comput Aided Geom Des 2019;71:190–201.
[21] Centin M, Signoroni A. Mesh denoising with (geo)metric fidelity. IEEE Trans Vis Comput Graphics 2018;24(8):2380–96.
[22] Wu C, Tai XC. Augmented Lagrangian method, Dual methods, and Split Bregman iteration for ROF, vectorial TV, and high order models. SIAM J Imaging Sci 2010;3(3):300–39.
[23] Zhang H, Wu C, Zhang J, Deng J. Variational mesh denoising using total variation and piecewise constant function space. IEEE Trans Vis Comput Graphics 2015;21(7):873–86.
[24] Liu Z, Lai R, Zhang H, Wu C. Triangulated surface denoising using high order regularization with dynamic weights. SIAM J Sci Comput 2019;41(1):1–26.
[25] Kim HS, Choi HK, Lee KH. Feature detection of triangular meshes based on tensor voting theory. Comput Aided Des 2009;41(1):47–58.
[26] Wei M, Liang L, Pang W-M, Wang J, Li W, Wu H. Tensor voting guided mesh denoising. IEEE Trans Autom Sci Eng 2016;14(2):931–45.
[27] Park MK, Lee SJ, Lee KH. Multi-scale tensor voting for feature extraction from unstructured point clouds. Graph Models 2012;74(4):197–208.
[28] Yadav SK, Reitebuch U, Skrodzki M, Zimmermann E, Polthier K. Constraint-based point set denoising using normal voting tensor and restricted quadratic error metrics. Comput Graph 2018;74:234–43.
[29] Zhang J, Cao J, Liu X, Wang J, Liu J, Shi X. Point cloud normal estimation via low-rank subspace clustering. Comput Graph 2013;37(6):697–706.
[30] Zhang J, Cao J, Liu X, Chen H, Li B, Liu L. Multi-normal estimation via pair consistency voting. IEEE Trans Vis Comput Graphics 2018;25(4):1693–706.
[31] Zheng Y, Li G, Wu S, Liu Y, Gao Y. Guided point cloud denoising via sharp feature skeletons. Vis Comput 2017;33(6–8):857–67.
[32] Zheng Y, Li G, Xu X, Wu S, Nie Y. Rolling normal filtering for point clouds. Comput Aided Geom Design 2018;62:16–28.